Context-Specific Independence in Directed Relational Probabilistic Models and its Influence on the Efficiency of Gibbs Sampling

Daan Fierens[1]

Abstract. There is currently a large interest in relational probabilistic models. While the concept of context-specific independence (CSI) has been well-studied for models such as Bayesian networks, this is not the case for relational probabilistic models. In this paper we show that directed relational probabilistic models often exhibit CSI by identifying three different sources of CSI in such models (two of which are inherent to the relational character of the models).

It is known that CSI can be used to speed up probabilistic inference. In this paper we show how to do this in a general way for approximate inference based on Gibbs sampling. We perform experiments on real-world data to analyze the influence of the three different types of CSI. The results show that exploiting CSI yields speedups of up to an order of magnitude.

1 INTRODUCTION

The main idea behind probabilistic models such as Bayesian networks is to exploit the independencies in a probability distribution in order to represent this distribution compactly. *Conditional independence (CI)* occurs when observing the state of some random variables makes some other random variables mutually independent. *Context-specific independence (CSI)* [2] occurs when the above does not hold regardless of what the observed state is, but only when it equals a particular state called the *context*. CSI is a weaker but more fine-grained notion of independence than CI [10]. CI is at the very core of probabilistic models such as Bayesian networks: the structure of a Bayesian networks reflects the conditional independencies in the associated probability distribution. CSI, in contrast, cannot be captured by the structure of a Bayesian network but only by its parameters (the conditional probability distributions) [2].

Independencies in probabilistic models are the key to making inference efficient. The fact that CSI is a weaker form of independence than CI has two opposing effects in this respect. On the one hand, making inference more efficient by exploiting CSI is more difficult than by exploiting CI. On the other hand, there is more often an opportunity to exploit CSI than to exploit CI, simply because CSI occurs more often (as it requires fewer independence assumptions).

In the field of artificial intelligence there is a large interest in *relational probabilistic models* [6, 3]. In this paper we focus on *directed* relational probabilistic models such as relational or first-order logical extensions of Bayesian networks [3, 6, 9, 5]. While CSI has been well-studied for propositional models such as Bayesian networks (e.g. [2, 10]), this is not the case for relational models. The *goal of this paper* is to study the presence of CSI in directed relational probabilistic models and the influence of CSI on the efficiency of inference with such models. The *contributions of this paper* are three-fold. First, we show that directed relational probabilistic models often exhibit CSI by identifying three different sources of CSI in such models, two of which are inherent to the relational character of the models (Section 3). Second, we show how to exploit CSI to make inference with such models more efficient, in particular for inference algorithms based on Gibbs sampling (Section 4). Third, we perform experiments on real-world data to validate the previous contributions. Our experimental results show that exploiting CSI in Gibbs sampling yields speedups of up to an order of magnitude and that these speedups grow with the size of the data (Section 5).

A variety of languages or 'formalisms' for representing directed relational probabilistic models has been proposed [3, 6, 9]. In this paper we use Logical Bayesian Networks [5], but our work applies to many of the other formalisms as well (see Section 3.3).

We now first discuss the idea behind directed relational probabilistic models and the formalism Logical Bayesian Networks.

2 DIRECTED RELATIONAL PROBABILISTIC MODELS

We assume familiarity with the basics of probability theory [8, Ch.1]. We denote random variables (RVs) by capital letters (for instance X), values or states of an RV by lower case letters (x), sets of RVs by boldface capital letters ($\mathbf{X}$) and their joint states by boldface lower case letters ($\mathbf{x}$). We refer to the set of possible states of an RV X as the *range* of X, denoted $range(X)$. We only consider *discrete* RVs (i.e. RVs with a finite range). A *conditional probability distribution (CPD)* for an RV X conditioned on a set of other RVs $\mathbf{Y}$ is a function that maps each state of $\mathbf{Y}$ to a probability distribution for X.

A *Bayesian network* [8] for a set of RVs $\mathbf{X}$ consists of a set of CPDs: for each $X \in \mathbf{X}$ there is a CPD for X conditioned on a (possibly empty) set of RVs called the *parents* of X. The probability distribution for X conditioned on its parents $\mathbf{pa}(X)$ is denoted $P(X \mid \mathbf{pa}(X))$. A Bayesian network represents a probability distribution $P(\mathbf{X})$ on the set of possible states of $\mathbf{X}$: $P(\mathbf{X})$ is the product of the CPDs in the Bayesian network, $P(\mathbf{X}) = \prod_{X \in \mathbf{X}} P(X \mid \mathbf{pa}(X))$. $P(\mathbf{X})$ is a proper probability distribution provided that the parent relation is acyclic (the parent relation is often visualized as a directed acyclic graph but given the CPDs this graph is redundant).

From a logical perspective, Bayesian networks use a propositional representation. It has been proposed many times to lift them to a first-order or relational representation, leading to many different for-

[1] Department of Computer Science, Katholieke Universiteit Leuven, Belgium, email: daan.fierens@cs.kuleuven.be

malisms for representing the resulting models [3, 6, 9]. Below we focus on the formalism of *Logical Bayesian Networks* [5, 4] (for the relationship to other formalisms, see Section 3.3). We only discuss the aspects of this formalism that are relevant to this paper.

The core idea behind Logical Bayesian Networks and many other related formalisms is to use *parameterized RVs* instead of regular RVs [4, 9]. Parameterized RVs have a number of typed parameters ranging over certain populations. For instance, to indicate whether a movie is blockbuster we can use a parameterized RV $blockb(M)$ and to indicate whether an actor acts in a movie we can use a parameterized RV $acts(A, M)$, with M and A parameters from the populations of movies and actors respectively. When each parameter in a parameterized RV is instantiated to a particular element of its population, we obtain a regular or *"concrete"* RV, for instance $blockb(m)$ or $acts(a, m)$, with m a particular movie and a a particular actor. Note that we denote parameters by upper case letters (A, M) and elements from a population by lower case letters (a, m).

2.1 Syntax of Logical Bayesian Networks (LBNs)

An LBN consists of a set of parameterized CPDs, one for each parameterized RV. Each CPD is represented as a relational or first-order logical *probability tree* [4, 5]. This is a decision tree in which each internal node contains a boolean test and each leaf contains a probability distribution on the range of the parameterized RV. Figure 1 shows the CPD for $blockb(M)$: it specifies that whether M is a blockbuster depends on the language of M and the number of popular actors in M. Parameters other than M are called *free* parameters in this CPD (e.g. A). Two types of tests can be used in the internal nodes of a tree.

- A test of the form $X = x$, with X a parameterized RV without free parameters (e.g. $language(M) = english$ in Figure 1).
- An *aggregate test*: a boolean test on the value of an aggregate function. In this paper we consider two aggregate functions.
  - *Count* (CNT) takes as argument a conjunction called the *collect-conjunction*. Each conjunct is of the form $X = x$ with X a parameterized RV that contains free parameters. The output of CNT is computed by collecting all different joint instantiations of the free parameters for which the collect-conjunction is satisfied, and counting them (e.g. for the test in Figure 1 this is the number of actors who act in M and have a high popularity).
  - *Mode* takes two arguments: a collect-conjunction and a parameterized RV called the *collect-RV*. The output is computed by collecting the state of the collect-RV for all different joint instantiations of the free parameters for which the collect-conjunction is satisfied, and then taking the mode (i.e. the most frequently occurring state). For instance, the test $\text{MODE}(acts(A, M) = yes, pop(A)) = high$ collects the popularity of all actors in M and checks whether the mode is 'high'.

2.2 Semantics of Logical Bayesian Networks

An LBN, in combination with a given population for each type, specifies a probability distribution. Let $\mathbf{X}$ denote the set of all concrete RVs that can be obtained by grounding all parameterized RVs with respect to the populations. The probability distribution specified by an LBN is a distribution on the set of possible states of $\mathbf{X}$: $P(\mathbf{X}) = \prod_{X \in \mathbf{X}} P(X|\mathbf{pa}(X))$, with $P(X|\mathbf{pa}(X))$ the CPD for X.

The CPD for a concrete RV $X \in \mathbf{X}$ is obtained from the parameterized CPD C_p for the parameterized RV X_p associated to X: we simply instantiate all parameters of X_p in C_p. For instance, to obtain the CPD for $blockb(m)$ we instantiate all occurrences of M in the parameterized CPD for $blockb(M)$ to m. Note that the only parameters that remain are those inside aggregate tests, in this case A.

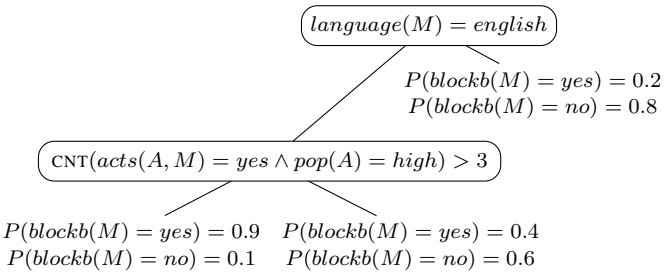

Figure 1. The CPD tree for the parameterized RV $blockb(M)$. When a test in a node succeeds the left branch is taken, otherwise the right branch.

The parents of a concrete RV X are all the concrete RVs in $\mathbf{X}$ that unify with an RV in an internal node of the CPD for X. For instance, the parents of $blockb(m)$ are $language(m)$ and an RV $acts(a, m)$ and $pop(a)$ for each a in the population of actors.

3 CONTEXT-SPECIFIC INDEPENDENCE IN LOGICAL BAYESIAN NETWORKS

Context-specific independence (CSI) is defined as follows [2].

Definition 1 (CSI) *Let X and Y be distinct RVs and $\mathbf{Z}$ be a set of RVs. Then X is* contextually independent *of Y given $\mathbf{Z} = \mathbf{z}$ if $P(X \mid Y, \mathbf{Z} = \mathbf{z}) = P(X \mid \mathbf{Z} = \mathbf{z})$ whenever $P(Y, \mathbf{Z} = \mathbf{z}) > 0$. This contextual independence is denoted $X \perp_c Y \mid \mathbf{Z} = \mathbf{z}$.*

It follows from the usual definition of conditional independence (CI) [2] that X is conditionally independent of Y given $\mathbf{Z}$ only if *for each possible state* $\mathbf{z}$, X is contextually independent of Y given $\mathbf{Z} = \mathbf{z}$. This shows that CSI is a more fine-grained notion than CI.

We now show that LBNs exhibit CSI. We only discuss *local CSI*, CSI that involves only an RV and its parents [2]. Non-local CSI can be derived from local CSI using the criterion of CSI-separation [2].

3.1 Sources of CSI in Logical Bayesian Networks

Local CSI occurs if an RV X is contextually independent of one of its parents Y given the state of some of its other parents. Since Y is a parent of X, we known that Y occurs at least once in the internal nodes of the CPD tree for X. In this section we assume that Y occurs exactly once. If Y occurs multiple times, see Section 3.2.

For Bayesian networks, it is known that local CSI can be derived from the CPDs [2]. Below we show that the same holds for LBNs. Concretely, we identify three sources of local CSI in LBNs.

3.1.1 Tree-CSI

The first source of CSI in LBNs is the tree structure of the CPDs. We refer to this type of CSI as *tree-CSI*. Let X be a concrete RV and let Y be a parent of X that occurs in the test in an internal node n of the CPD tree for X. There are two possible cases.

- *n is the root node of the CPD tree:*
  In this case there is no tree-CSI between X and Y.

- *n is not the root node of the CPD tree:*
  Let $path(n)$ be the condition describing the path from the root to n [2]. X is contextually independent of Y given that $path(n)$ is false: $X \perp_c Y \mid \neg path(n)$. For instance, if we use the CPD of Figure 1, it holds for any movie m and actor a in the population that $blockb(m) \perp_c pop(a) \mid \neg(language(m) = english)$.

Tree-CSI has been well-studied in the context of Bayesian networks [2]. Hence the above result is not our contribution. In contrast, the next two types of CSI are inherent to the relational character of LBNs and have (to the best of our knowledge) not been studied before.

3.1.2 Conjunction-CSI

The second source of CSI in LBNs are the conjunctions inside aggregate tests. We refer to this type of CSI as *conjunction-CSI*. Let X be a concrete RV and let Y be a parent of X that occurs inside an aggregate test in the CPD tree for X. There are three possible cases.

- *Y occurs in the collect-conjunction φ of a (count or mode) aggregate and φ has only one conjunct:*
  In this case there is no conjunction-CSI between X and Y.
- *Y occurs in the collect-conjunction φ of a (count or mode) aggregate and φ has two or more conjuncts:*
  In general, φ is of the form $\wedge_i(Z_i = z_i)$. Suppose that Y occurs in the j-th conjunct (i.e. Y matches with Z_j). Let $F_1, \ldots F_l$ be the parameters that are free but do not occur in Y. Then it holds that $X \perp_c Y \mid \neg(\exists F_1, \ldots F_l : \wedge_{i \neq j} Z_i = z_i)$. Let us explain this further with some examples. For the count aggregate in Figure 1 it holds that $blockb(m) \perp_c pop(a) \mid \neg(acts(a, m) = yes)$. In this case no existential quantifier is needed. An example of where it is needed is the following. Suppose that the CPD tree for an RV $p(A)$ contains the aggregate $\text{CNT}(q(A, B) = yes \wedge r(B, C) = yes \wedge s(C) = v)$. Then it holds that $p(a) \perp_c s(c) \mid \neg(\exists B : q(a, B) = yes \wedge r(B, c) = yes)$. The rationale is as follows: any instantiation of B for which $q(a, B) = yes \wedge r(B, c) = yes$ is true, creates a probabilistic influence of $s(c)$ on $p(a)$. Hence there only is CSI if there exists no such instantiation of B.
- *Y matches with the collect-RV of a mode aggregate:*
  Let φ be the collect-conjunction of the mode aggregate and let $F_1, \ldots F_l$ be the parameters that are free but do not occur in Y. Then it holds that $X \perp_c Y \mid \neg(\exists F_1, \ldots F_l : \varphi)$. For instance, if the CPD for $blockb(M)$ would contain an aggregate $\text{MODE}(acts(A, M) = yes, pop(A))$, then it would hold that $blockb(m) \perp_c pop(a) \mid \neg(acts(a, m) = yes)$. The reasoning is very similar to that of the second case in this list.

3.1.3 Aggregate-CSI

The third source of CSI in LBNs are the aggregate functions. We refer to this type of CSI as *aggregate-CSI*. The formal specification of the conditions under which aggregate-CSI occurs is rather verbose and does not fit in the space available (nor does it provide a lot of intuition). Below we explain the main idea. This is sufficient for dealing with all the real-world LBNs that we use in our experiments.

- *Count aggregates:*
  Consider the test in Figure 1: $\text{CNT}(acts(A, M) = yes \wedge pop(A) = high) > 3$. It holds that $blockb(m) \perp_c pop(a) \mid (\text{CNT}_{A \neq a}(acts(A, m) = yes \wedge pop(A) = high) \neq 3)$. Here $\text{CNT}_{A \neq a}()$ means the usual except that we do not count instantiations in which $A = a$. The intuition behind this CSI-condition is the following. $\text{CNT}()$ can be computed as $\text{CNT}_{A \neq a}() + \Delta$, with Δ being 1 if $acts(a, m) = yes \wedge pop(a) = high$ is true and 0 otherwise. The test checks whether $\text{CNT}() > 3$. Whether this test succeeds or not depends on Δ only if $\text{CNT}_{A \neq a}()$ is exactly 3. If $\text{CNT}_{A \neq a}()$ is greater than 3, the test succeeds regardless of Δ, and hence regardless of $pop(a)$. If $\text{CNT}_{A \neq a}()$ is smaller than 3, the test fails regardless of $pop(a)$. This proves the above CSI-condition.
- *Mode aggregates:*
  Suppose that the CPD for $blockb(M)$ would contain a test $\text{MODE}(acts(A, M) = yes, pop(A))$=$high$. Then it holds that $blockb(m) \perp_c pop(a) \mid (\text{MODE\_MARGIN}^{high}_{A \neq A}(acts(A, m) = yes, pop(A)) \notin \{0, -1\})$. Here $\text{MODE\_MARGIN}^{high}()$ is the number of times that the collected state is 'high' minus the number of times that it is val, with val the most frequent state apart from 'high'. The intuition is similar as for CNT: if the margin is greater than 0, the above test succeeds regardless of $pop(a)$; if the margin is smaller than -1, the test fails regardless of $pop(a)$.

3.2 Putting it all together

So far we have considered the three types of CSI separately. Any of the three types is sufficient for making X contextually independent of Y. For instance, we have seen above that $blockb(M)$ can be contextually-independent of $pop(A)$ due to any of the three types. In general, it holds that $X \perp_c Y \mid (c^{TREE} \vee c^{CONJ} \vee c^{AGGR})$, where c^{TREE} denotes the condition under which X is contextually independent of Y due to tree-CSI and similar for c^{CONJ} and c^{AGGR}.

So far we also assumed that Y occurs only once in the internal nodes of the CPD tree for X. If Y occurs multiple times (for instance in different branches of the tree), then X is only contextually independent of Y if there is CSI for each of the occurrences. To be precise, it holds that $X \perp_c Y \mid \wedge_i (c_i^{TREE} \vee c_i^{CONJ} \vee c_i^{AGGR})$, where the conjunction $\wedge_i$ ranges over all occurrences of Y, and c_i^{TREE} is the condition under which X is contextually independent of the i-th occurrence of Y due to tree-CSI and similar for c_i^{CONJ} and c_i^{AGGR}.

3.3 Relevance of our results

Above we focussed on LBNs. Our discussion is also relevant to many other formalisms for representing relational probabilistic models.

We identified three types of CSI in LBNs. *Conjunction-CSI* occurs for virtually any relational probabilistic formalism, as conjunctions of logical literals or relations are used in some way in all of them. *Aggregate-CSI* of course only occurs in those formalisms that use the concept of aggregate tests. This includes several well-known formalisms like Probabilistic Relational Models [6, Ch.5], Bayesian Logic Programs [6, Ch.10], CLP($\mathcal{BN}$) [3, Ch.6], and probabilistic logic programming formalisms that allow the use of meta-predicates [4]. *Tree-CSI* is, unlike the previous types of CSI, not inherent to the relational character of LBNs but to the tree-structured CPDs that we use in LBNs. Tree-CSI has limited relevance to other relational probabilistic formalisms: the only other formalisms that use tree-structured CPDs are BLOG (to some extent [7]) and Relational Dependency Networks (which deals with undirected models [6, Ch.8]).

4 EXPLOITING CONTEXT-SPECIFIC INDEPENDENCE IN GIBBS SAMPLING

We now consider the task of performing probabilistic inference with an LBN. Given the set of all concrete RVs $\mathbf{X}$ (as determined by

```
procedure GIBBS_SAMPLING(E, Q, D)
 1  for each E ∈ E                      8  repeat until enough samples
 2      set E to its known state        9      for each U ∈ U
 3  U = Q ∪ D                           10         compute P_MB(U)
 4  for each U ∈ U                      11         sample u from P_MB(U)
 5      set U to random state           12         set U to u
 6      if U ∈ Q                        13         if U ∈ Q
 7          init counters for U         14             update counters for U
```

Figure 2. The Gibbs sampling algorithm.

the populations for each type) an LBN defines a probability distribution $P(\mathbf{X})$. In a typical inference task, we have certain evidence, i.e. we know the true state of a subset of $\mathbf{X}$, and we need to answer certain questions about the distribution $P(\mathbf{X})$ conditioned on this evidence. The most common inference task is to compute marginal probabilities (a marginal probability is the probability that a particular non-evidence RV is in a particular state given the evidence).

Inference with relational probabilistic models such as LBNs is often computationally intractable for real-world population sizes (inference with Bayesian networks is NP-hard [8]). Hence, one often uses approximate inference algorithms such as Monte Carlo algorithms that draw samples from the given distribution conditioned on the evidence. A very popular such algorithm is *Gibbs sampling*.

4.1 Gibbs sampling

Let $\mathbf{E}$ be the set of evidence RVs, $\mathbf{Q}$ the set of *query RVs* (for which we need to compute marginal probabilities) and $\mathbf{D}$ the set of remaining RVs, also called *don't care RVs* ($\mathbf{D} = \mathbf{X} \setminus \mathbf{Q} \setminus \mathbf{E}$). We call an RV *unobserved* if it is in $\mathbf{Q} \cup \mathbf{D}$. Pseudocode for the Gibbs sampling algorithm [1, 11] is shown in Figure 2. We now explain this further.

First all evidence RVs are instantiated to their known state and all unobserved RVs are instantiated to a random state. We also create a number of counters: for each query RV Q and each $q \in range(Q)$ we create a counter to store the number of samples in which Q is in state q. Next the sampling process starts. To create one sample, we visit (in an arbitrary but fixed order) all unobserved RVs. For each visited RV U, we compute the distribution $P_{MB}(U)$ for U conditioned on the current state of all RVs in the Markov blanket of U (see below), we randomly sample a state u from $P_{MB}(U)$, we set U to u, and if U is a query RV we increment the counter for $\langle U, u \rangle$. This entire procedure is repeated until enough samples have been collected. In practice, we use a slight variation of the above approach which includes a number of common optimizations.[2]

After having drawn a number of samples, it is straightforward to construct an estimate of all required marginal probabilities based on the computed counts: the estimated marginal probability that a query RV Q is in a given state is equal to the number of samples in which Q is in that state divided by the total number of samples N. The higher N, the closer the estimated marginal probabilities will generally be to their correct values (with N going to infinity the estimates converge to the correct values provided that all CPDs are strictly positive) [1].

[2] We check for each query RV Q whether all RVs in its Markov blanket are evidence RVs. If yes, we can compute the marginal of Q directly and we discard Q in the sampling process (i.e. we do not visit it). We also check for each don't care RV whether it is relevant to the query RVs. We do this by computing the 'support network' [3, Ch.7] for the query RVs. All don't care RVs that do not occur in this network are independent of the query RVs given the evidence, hence we can discard them in the sampling process.

4.2 The need for efficient Gibbs sampling

Gibbs sampling is often used by giving the sampling process a fixed time to run before computing the estimates. In this case, any gain in efficiency of the sampling process might lead to a gain in accuracy of the estimates (the less time per sample, the more samples in the given time, so the more accurate the estimates). Hence it is important that the sampling process is as efficient as possible.

The bottleneck in Gibbs sampling is the computation of the distribution $P_{MB}(U)$ of an RV U conditioned on its Markov blanket (line 10 in Figure 2). This distribution is the following [1]

$$P_{MB}(U) = c\, P(U \mid \mathbf{pa}(U)) \prod_{X \in \mathbf{ch}(U)} P(X \mid \mathbf{pa}(X)),$$

with c a normalization constant and $\mathbf{ch}(U)$ the set of children of U (X is called a child of U if and only if U is a parent of X). Finding this distribution requires a number of computations, mainly applying or "calling" a number of CPDs. Concretely, we call the CPD for U (given the current state of U's parents) and then we loop over all $u' \in range(U)$ and for each u' we set U to u' and we call the CPD for each of the children of U (given the current state of their parents, which includes u'). At the end, we compute the normalization constant c, and we obtain the distribution $P_{MB}(U)$.

The above shows that computing $P_{MB}(U)$ requires $1 + |range(U)| \times |\mathbf{ch}(U)|$ different CPD-calls. This can be computationally expensive since an RV in a relational model can have many children (e.g. several dozens) and even a single CPD-call can be relatively expensive when the CPD involves aggregate tests. Moreover, computing a distribution $P_{MB}(U)$ needs to happen millions of times during the entire Gibbs sampling process. Hence, eliminating any redundant CPD-calls that might occur during Gibbs sampling can yield significant speedups. This is where CSI comes into play.

4.3 The influence of CSI on Gibbs sampling

Consider what happens in the computation of $P_{MB}(U)$ when there is CSI, concretely when some child X of U is contextually independent of U given the current state of the other RVs. Because of the CSI, the factor $P(X \mid \mathbf{pa}(X))$ will be the same for all states u' of U, so in the above normalization step these factors will cancel out. In other words: a child that is contextually independent of U does not have any actual influence on the distribution $P_{MB}(U)$.

Instead of taking into account all children of U in the computation of $P_{MB}(U)$, we can limit ourselves to those children that are not contextually independent of U. This saves us a number of CPD-calls, concretely $|range(U)|$ CPD-calls per child that is contextually independent. To find the children of U that are not contextually independent of U we simply loop over all children, and for each child X we check whether there is CSI between X and U given the current state of the other RVs. The exact condition under which there is CSI between an RV and its child is determined as in Section 3. We refer to this approach as *Gibbs sampling with CSI*.[3]

We have defined three different types of CSI in Section 3. We can in principle decide for each type of CSI separately whether we exploit it or not. Recall that the CSI-condition is generally of the

[3] Our Gibbs sampling algorithm can also be used for other formalisms (for directed relational probabilistic models) than LBNs. The algorithm has access to the model at hand only through two functions: one for calling the CPD for an RV and one for finding the children of an RV U that are not contextually independent of U. To do Gibbs sampling with a model we only need to supply these two functions, no matter what the formalism is.

form $\wedge_i(c_i^{TREE} \vee c_i^{CONJ} \vee c_i^{AGGR})$. If we decide, for instance, not to exploit conjunction-CSI, then we simply check the condition $\wedge_i(c_i^{TREE} \vee c_i^{AGGR})$ instead. Since we considered 3 different types of CSI, there are 8 (=2^3) possible combinations. In other words, there are 8 different CSI-settings that we can run Gibbs sampling with, ranging from not exploiting CSI (standard Gibbs sampling) to exploiting all types of CSI. Note that these different CSI-settings all produce exactly the same sequence of samples (because with each CSI-setting we obtain the same, correct, distribution $P_{MB}(U)$).

It is not necessarily the case that the more different types of CSI we exploit, the more efficient Gibbs sampling will be. This is because exploiting CSI has two opposing effects. On the one hand, CSI reduces the number of required CPD-calls. The more different types of CSI we exploit, the higher the savings. On the other hand, checking for each child whether the CSI-condition holds also consumes time. The more different types of CSI we exploit, the more complex the CSI-condition becomes, and hence the more time it takes to check it. Hence, whether exploiting a particular kind of CSI makes Gibbs sampling faster or not depends on which of these two effects is dominant. We perform experiments to investigate this.

5 EXPERIMENTS

The aim of our experiments is to analyze the influence of the different types of CSI on the efficiency of Gibbs sampling. Recall that we can run Gibbs sampling with 8 different CSI-settings. We perform experiments on a number of inference tasks and compare the runtimes of Gibbs sampling with each of these 8 settings.

5.1 Experimental setup

We use three real-world datasets that are very common in the area of relational probabilistic models: IMDB, UWCSE and WebKB [3]. To obtain the model (the LBN) for a dataset, one approach could have been to construct the model manually. However, we would then have to choose ourselves to which extent the model exhibits CSI, which could of course greatly influence our results. Hence we instead took a more objective approach: we used machine learning algorithms to learn a model from each dataset (we used the ordering-search algorithm for LBNs [5]). Some statistics about the datasets and the models are given in Table 1 (see Fierens et al. [5] for more information).

Table 1. Properties of the datasets (number of RVs) and the corresponding models (types of aggregate tests).

Dataset	Parameterized RVs	Concrete RVs	Aggregates
IMDB	7	2852	Count
UWCSE	10	9607	Count + Mode
WebKB	5	3194	Count

We use two types of inference tasks. In the first type, all concrete RVs associated to a particular parameterized RV X are query RVs and all others are evidence RVs (this is a common setting [11]). For each parameterized RV X in each dataset there is a separate inference task. In more than half of the resulting inference tasks the entire Markov blanket of each query RV is given as evidence; we omit such tasks since they are trivial (they do not require Gibbs sampling). In the second type of inference tasks, we randomly select for each dataset 25% of all concrete RVs as query RVs and we use all others as evidence RVs. We perform each experiment 10 times, each time with a different random selection, and we report the average runtime. We also performed experiments with other percentages of query RVs (5%, 15%, 35% and 50%), the results are very similar to those with 25% (when we use more query RVs, the runtime goes up for all CSI-settings but the relative differences between these settings stay quasi constant). In total we use 11 different inference tasks, see Table 2.

Table 2. Definition of the 11 inference tasks and the runtime of standard Gibbs sampling (i.e. without CSI) on these tasks.

Task	Dataset	Query RVs	Time	Task	Dataset	Query RVs	Time
T1	IMDB	*acts*	1249s	T7	UWCSE	*teaches*	3880s
T2	IMDB	*directs*	258s	T8	UWCSE	random 25%	3031s
T3	IMDB	random 25%	406s	T9	WebKB	*has-project*	1464s
T4	UWCSE	*advised-by*	4162s	T10	WebKB	*prof*	1528s
T5	UWCSE	*co-author*	566s	T11	WebKB	random 25%	819s
T6	UWCSE	*phase*	535s				

We measured the runtime of Gibbs sampling with the 8 different CSI-settings on the 11 inference tasks. We performed Gibbs sampling with 10 parallel chains and collected 11000 samples (the first 1000 samples are used as burn-in samples). Since our main goal is to investigate the *relative* efficiency of the different CSI-settings, the choice of the number of samples does not heavily influence our conclusions. Recall that for a given task all CSI-settings produce exactly the same sequence of samples (Section 4.3).

5.2 Experimental results

Table 2 shows the runtime for the setting without CSI (standard Gibbs sampling). Table 3 shows the speedup factor for the other settings (due to space limits we can only show 4 of the 7 other settings). The speedup factor for a CSI-setting S is the runtime of standard Gibbs sampling divided by the runtime of S. We now analyze our results by answering three questions that successively come to mind.

Table 3. Speedup factors for various CSI-settings (ALL-EXC-TREE stands for ALL-EXCEPT-TREE, and similar for the others).

Task	ALL	ALL-EXC-TREE	ALL-EXC-CONJ	ALL-EXC-AGGR	Task	ALL	ALL-EXC-TREE	ALL-EXC-CONJ	ALL-EXC-AGGR
T1	4.56	4.61	2.70	1.15	T7	6.17	6.18	0.81	7.35
T2	2.31	2.26	0.88	2.74	T8	3.36	2.58	0.96	3.43
T3	3.32	3.18	1.90	1.30	T9	2.96	2.44	2.96	1.19
T4	5.45	5.31	0.83	6.25	T10	1.38	1.31	1.38	1.06
T5	1.13	0.99	1.15	0.99	T11	1.83	1.68	1.83	1.11
T6	13.02	10.64	1.43	9.29	*AVG*	*4.62*	*4.22*	*3.26*	*3.75*

5.2.1 *Does CSI help to speed up Gibbs sampling?*

Let us first consider the CSI-setting in which all types of CSI are exploited, denoted ALL in Table 3. The speedup depends heavily on the task at hand: in the worst case it is negligible (a factor 1.13 on T5), in the best case it is large (a factor 13 on T6). For the tasks where the speedup is negligible, we found that there is simply almost no CSI in the relevant CPDs. On average the speedup is a factor 4.62 so overall CSI is quite useful to speed up Gibbs sampling.

5.2.2 *Which types of CSI are responsible for the speedups?*

Let us first consider the influence of *tree-CSI*. To do so, we can compare the speedup for ALL and ALL-EXCEPT-TREE in Table 3. The gap

between the two is usually quite small. This shows that tree-CSI has only a small contribution to the speedup of ALL. Indeed, we noticed that in the computation of $P_{MB}(U)$ it happens relatively rarely that a child of U is contextually independent of U due to tree-CSI. This is because our CPDs are mostly shallow trees, and the same RVs often occur in several branches of the tree, which inhibits tree-CSI.

Let us now consider *conjunction-CSI*. The gap between the speedups for ALL and ALL-EXCEPT-CONJ is often quite large, which indicates that conjunction-CSI has a large contribution to the speedup of ALL. To verify this we performed additional experiments in which we varied the population size for the tasks on which conjunction-CSI has the largest contribution. Figure 3 (top, left+right) shows some representative results: the speedup grows with the population size and is almost entirely due to conjunction-CSI on these tasks.

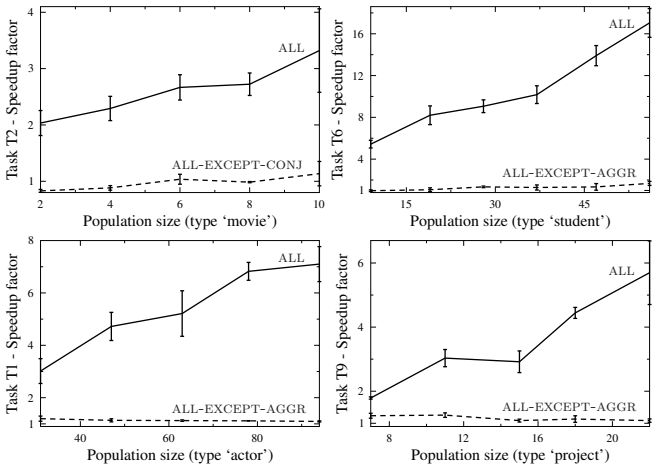

Figure 3. Influence of the population size on the speedup (each result is an average over three measurements, error bars indicate the standard deviation).

Similarly, there are other tasks on which *aggregate-CSI* is the main responsible for the speedups. Figure 3 (bottom, left+right) shows some tasks for which this is the case. The speedup factor again grows with the population size. This is a positive result: speedups are more necessary for large population sizes than for small ones.

To summarize, mainly the two types of CSI that are inherent to the relational character of LBNs are responsible for the speedups.

5.2.3 What is the optimal (most efficient) CSI-setting?

Table 4 shows the optimal setting for each of the tasks. On 5 of the 11 tasks, ALL is the optimal setting. On the other tasks, ALL is outperformed by a setting that exploits fewer types of CSI. This shows that exploiting a particular type of CSI sometimes yields a slowdown instead of speedup. This happens when the cost of detecting that type of CSI (checking the CSI-condition) outweighs the resulting gain (the reduced number of CPD-calls). This raises the question: when we are given a new inference task, which CSI-setting should we use?

Table 4 shows that when ALL is not the optimal setting, which setting is optimal instead depends heavily on the task at hand. Given a new inference task, we currently do not know how to predict on beforehand which setting will be the optimal one. Hence, using ALL is the safest approach. Moreover, Table 4 shows that even if we were able to predict the optimal setting for a task, the speedup with respect to ALL would be very modest at best (a factor of 1.22 in the very best case). Hence our recommendation is to always use the setting ALL.

Table 4. The optimal (most efficient) CSI-setting for each task.

Task	Optimal setting	Improvement w.r.t. ALL	Task	Optimal setting	Improvement w.r.t. ALL
T1	ALL-EXC-TREE	1.01	T7	ONLY-CONJ	1.19
T2	ONLY-CONJ	1.22	T8	ALL-EXC-AGGR	1.02
T3	ALL	-	T9	ALL	-
T4	ALL-EXC-AGGR	1.15	T10	ALL	-
T5	ALL-EXC-CONJ	1.02	T11	ALL	-
T6	ALL	-			

6 CONCLUSION

We studied the presence of context-specific independence (CSI) in directed relational probabilistic models. We showed that such models often exhibit CSI by identifying three different sources of CSI in such models. We then considered the task of performing approximate inference with such models using Gibbs sampling. We argued that it is important that the sampling process is as efficient as possible and showed how CSI can be exploited to this end. Experiments on real-world data show that this yields speedups of up to an order of magnitude. The speedups often grow with the population size and are mainly due to the two types of CSI that are inherent to the relational character of the models, namely conjunction-CSI and aggregate-CSI.

ACKNOWLEDGEMENTS

This research is supported by Research Foundation-Flanders (FWO Vlaanderen), GOA/08/008 'Probabilistic Logic Learning' and Research Fund K.U.Leuven.

REFERENCES

[1] B. Bidyuk and R. Dechter, 'Cutset sampling for Bayesian networks', *Journal of Artificial Intelligence Research*, **28**, 1–48, (2007).
[2] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller, 'Context-specific independence in Bayesian networks', in *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 115–123. Morgan Kaufmann, (1996).
[3] L. De Raedt, P. Frasconi, K. Kersting, and S. Muggleton, *Probabilistic Inductive Logic Programming*, Springer, 2008.
[4] D. Fierens, 'On the relationship between logical Bayesian networks and probabilistic logic programming based on the distribution semantics', in *Proceedings of the 19th International Conference on Inductive Logic Programming (ILP)*, volume 5989 of *Lecture Notes in Computer Science*. Springer, (2009). In press.
[5] D. Fierens, J. Ramon, M. Bruynooghe, and H. Blockeel, 'Learning directed probabilistic logical models: Ordering-search versus structure-search', *Annals of Mathematics and Artificial Intelligence*, **54**(1), 99–133, (2008).
[6] L. Getoor and B. Taskar, *An Introduction to Statistical Relational Learning*, MIT Press, 2007.
[7] B. Milch and S. Russell, 'General-purpose MCMC inference over relational structures', in *Proceedings of the 22nd Conference in Uncertainty in Artificial Intelligence (UAI)*, pp. 349–358. AUAI Press, (2006).
[8] R.E. Neapolitan, *Learning Bayesian Networks*, Prentice Hall, 2003.
[9] D. Poole, 'First-order probabilistic inference', in *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 985–991. Morgan Kaufmann, (2003).
[10] D. Poole and N.L. Zhang, 'Exploiting contextual independence in probabilistic inference', *Journal of Artificial Intelligence Research*, **18**, 263–313, (2003).
[11] H. Poon and P. Domingos, 'Sound and efficient inference with probabilistic and deterministic dependencies', in *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, pp. 214–225. AAAI Press, (2006).